From Optimization to Control: Quasi Policy Iteration

Mohammad Amin Sharifi Kolarijani and Peyman Mohajerin Esfahani

Abstract. Recent control algorithms for Markov decision processes (MDPs) have been designed using an implicit analogy with well-established optimization algorithms. In this paper, we make this analogy explicit across four problem classes with a unified solution characterization. This novel framework, in turn, allows for a systematic transformation of algorithms from one domain to the other. In particular, we identify equivalent optimization and control algorithms that have already been pointed out in the existing literature, but mostly in a scattered way. With this unifying framework in mind, we then exploit two linear structural constraints specific to MDPs for approximating the Hessian in a second-order-type algorithm from optimization, namely, Anderson mixing. This leads to a novel first-order control algorithm that modifies the standard value iteration (VI) algorithm by incorporating two new directions and adaptive step sizes. While the proposed algorithm, coined as quasi-policy iteration (QPI), has the same computational complexity as VI, it interestingly exhibits an empirical convergence behavior similar to policy iteration with a very low sensitivity to the discount factor. We further extend QPI to the model-free setting (i.e., reinforcement learning), to which we refer as the quasi-policy learning (QPL), and guarantee its convergence by a novel safeguarding technique. QPL also shares similar features with QPI in the sense that despite the first-order per-iteration complexity, its convergent behavior and sensitivity to discount factor are comparable with the second-order algorithms such as Zap Q-learning.

Keywords: Dynamic programming, reinforcement learning, optimization algorithms, quasi-Newton methods, Markov decision processes.

1. Introduction

The problem of control, or the decision-making problem as it is also known within the operations research community, has been the subject of much research since the introduction of the Bellman principle of optimality in the late 1950s [4]. Apart from the fact that policy iteration is an instance of the Newton method, which has been known since the late 1970s [41], more recent works have made implicit use of the relationship between optimization and control problems to develop new control algorithms, with faster convergence and/or lower complexity, inspired by their counterparts

Date: November 21, 2023.

The authors are with Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. Email: {M.A.SharifiKolarijani, P.MohajerinEsfahani}@tudelft.nl.

This research is part of a project that has received funding from the European Research Council (ERC) under the grant TRUST-949796.

for solving optimization problems. For instance, accelerated versions of value iteration in [19] are inspired by Polyak momentum and Nesterov acceleration in convex optimization, while the Q-learning combined with Polyak momentum and Nesterov acceleration produces momentum Q-learning [52].

The implicit connection between optimization algorithms and control algorithms for Markov decision processes (MDPs) with a finite state-action space has also been studied more systematically. In [50], the authors look at the connection between *constrained* convex optimization algorithms and control algorithms such as Frank-Wolfe algorithm [14] and conservative policy iteration [24]. A detailed comparison between *deterministic* optimization algorithms and *model-based*[1] control algorithm is also provided in [20], where the author looks at a wide range of optimization algorithms including gradient descent, accelerated gradient descent, Newton method, and quasi-Newton method and their counterparts for solving control problems.

When it comes to infinite (continuous) state-action spaces, except in special cases such as linear–quadratic regulators (LQR), one needs to resort to finite-dimensional approximation techniques for computational purposes. This approximation may be at the modeling level by aggregation (discretization) of the state and action spaces, which readily falls into the finite MDP setting mentioned above [6, 40]. Alternatively, one may directly approximate the value function via finite parametrization by minimizing (a proxy of) the residual of its fixed-point characterization based on the Bellman principle of optimality [7, 48]. Examples of such include linear parameterization [9, 49], or nonlinear parameterization with, for instance, neural network architectures [8, 46, 47] or max-plus approximation [5, 18, 29, 28, 34]. We also note that there is an alternative characterization of the original function as the solution to an infinite-dimensional linear program [21], paving the way for approximation techniques via finite tractable convex optimization [11, 22, 35]. With this view of the literature, it is worth noting that one can cast almost all of these approximation techniques as the solution to a finite-dimensional fixed-point or convex optimization problem.

Motivated by this observation, this study provides an explicit framework unifying the tight link between convex optimization (stochastic and deterministic, respectively) and optimal control (model-based dynamic programming and model-free reinforcement learning, respectively). Specifically, this goal is achieved by exploiting the (expected) root-finding characterization of optimization problems and the (expected) fixed-point characterization of control problems. This explicit equivalence relationship not only allows us to identify existing (and mostly known) equivalent algorithms for optimization and control, but also provides a concrete methodology for developing new algorithms in one domain based on the existing algorithms in the other. In particular, inspired by quasi-Newton method, we develop a new control algorithm, quasi-policy iteration, that employs an approximation of the second-order information to speed up the convergence with the same per-iteration computational cost as first-order methods

[1]In this paper, the terminologies of "model-free" and "mode-based" indicate the *available information (oracle)*, i.e., whether we have access to the model or only the system trajectory (samples); see Section 2.1 for more details. We note that this is different from the common terminologies in the RL literature where these terms refer to the *solution approach*, i.e., whether we identify the model along the way (model-based RL) or directly solve the Bellman equation to find the value function (model-free RL).

Contributions. The contributions of this paper are as follows:

(1) Optimization vs. control: We provide a framework by using the (expected) root-finding condition in convex optimization and the (expected) fixed-point condition in control problems that gives a novel unifying characterization for four different classes of problems with two types of oracles (available information) in these domains. The framework yields an explicit transformation of deterministic (stochastic) convex optimization problems to model-based (model-free) control problems, and vice versa (Table 1). These transformations, in turn, allow for a systematic transformation of algorithms from one domain to the other (Table 2).

(2) Quasi-policy iteration: Thanks to the connection above, we adopt the quasi-Newton method from convex and exploit the properties of Bellman operator optimization to introduce the quasi-policy iteration (QPI) algorithm with the update rule

$$v_{k+1} = (1 - \delta_k)T(v_k) + \delta_k c_k + \lambda_k \mathbf{1}, \tag{1}$$

where v_k is the value function at iteration k of the algorithm, T is the Bellman operator, c_k is the stage cost under the greedy policy w.r.t. v_k, and $\mathbf{1}$ is the all-one vector. The proposed QPI algorithm in (1) has the following distinct features:

(2a) **Hessian approximation via structural information**: QPI is based on a novel approximation of the "Hessian" matrix in the policy iteration (PI) algorithm by exploiting two linear structural constraints specific to Markov decision processes (MDPs) (Theorem 4.2).

(2b) **First-order directions with adaptive step-size**: QPI can be viewed as a modification of the standard value iteration (VI) $v_{k+1} = T(v_k)$ using the two novel directions $c_k - T(v_k)$ and $\mathbf{1}$ along with adaptive step-sizes δ_k and λ_k that depend on v_k.

(2c) **Convergence rate and sensitivity to discount factor**: The per-iteration computational complexity of QPI is the same as VI (i.e., $\mathcal{O}(n^2)$ where n is the number of states) and its linear convergence can be guaranteed by safeguarding via standard VI (Theorem 4.2). However, in our numerical simulations with randomly generated Garnet MDPs, QPI exhibits an empirical behavior similar to PI (which has an $\mathcal{O}(n^3)$ per-iteration complexity) concerning *convergence rate* and *sensitivity to the discounted factor* (Figure 1a).

(2d) **Extension to model-free control**: We also introduce the quasi-policy learning (QPL) algorithm, the stochastic version of QPI, as a novel model-free algorithm, and guarantee its convergence by safeguarding via standard Q-learning (QL) algorithm (Theorem 4.3). To the best of our knowledge, our proposed safeguarding technique is also new in the RL literature. While the proposed QPL has an $\mathcal{O}(n)$ per-iteration complexity similar to synchronous QL, in our numerical simulation with randomly generated Garnet MDPs, it shows a much faster convergent behavior and a sensitivity to discount factor comparable with synchronous Zap Q-learning [13] which has an $\mathcal{O}(n^2)$ per-iteration complexity (Figure 1b).

The paper is organized as follows. In Section 2, we describe the connection between optimization and control problems by providing the explicit transformations between them. We then use this framework to look at equivalent algorithms from the two domains in Section 3. In Section 4, we introduce and analyze the model-based QPI algorithm and its model-free extension, the QPL algorithm. The performance of these algorithms is then compared with multiple control algorithms via extensive numerical experiments in Section 5.

Notations. For a vector $v \in \mathbb{R}^n$, we use $v(i)$ and $[v](i)$ to denote its i-th element. Similarly, $M(i,j)$ and $[M](i,j)$ denote the element in row i and column j of the matrix $M \in \mathbb{R}^{m \times n}$. We use $\cdot^\top$ to denote the transpose of a vector/matrix. We use $\|\cdot\|_2$ and $\|\cdot\|_\infty$ to denote the 2-norm and ∞-norm of a vector, respectively. We use $\|\cdot\|_2$ and $\|\cdot\|_F$ for the induced 2-norm and the Frobenius norm of a matrix, respectively. Let $x \sim \mathbb{P}$ be a random variable with distribution $\mathbb{P}$. We particularly use $\hat{x} \sim \mathbb{P}$ to denote *a sample of the random variable* x drawn from the distribution $\mathbb{P}$. The identity operator is denoted by Id. We use $\mathbf{1}_n$ and $\mathbf{0}_n$ to denote the n-dimensional vectors of ones and zeros, respectively. With some abuse of notation, we denote the i-th unit vector by $\mathbf{1}_n(i)$, that is, the vector with its i-th element equal to 1 and all other elements equal to 0. We also use I_n and $E_n = \mathbf{1}_n \mathbf{1}_n^\top$ to denote the n-by-n identity and all-one matrices, respectively. We drop the subscript n when there is no confusion about the dimension.

2. Equivalence Transformations

In this section, we provide the generic framework that connects the optimization problems to the control problems. In particular, we provide the explicit transformations between specific characterizations of the solutions to these problems. Table 1 provides a condensed summary of this framework.

2.1. Control problem

A common formulation of the control problem relies on the concept of *Markov decision processes* (MDPs). MDPs are a powerful modeling framework for stochastic environments that can be controlled to minimize some measure of cost. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, c, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. The transition kernel $\mathbb{P}$ encapsulates the state dynamics: for each triplet $(s, a, s^+) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, it gives the probability $\mathbb{P}(s^+|s,a)$ of the transition to state s^+ given that the system is in state s and the chosen control is a. The cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, bounded from below, represents the cost $c(s,a)$ of taking the control action a while the system is in state s. The discount factor $\gamma \in (0,1)$ can be seen as a trade-off parameter between short- and long-term costs. *In this study, we consider tabular MDPs with a finite state-action space. In particular, we take* $\mathcal{S} = \{1, 2, \dots, n\}$ *and* $\mathcal{A} = \{1, 2, \dots, m\}$. This, in turn, allows us to treat functions $f : \mathcal{S} \to \mathbb{R}$ and $g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as vectors $f \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n$ and $g \in \mathbb{R}^{|\mathcal{S}| \cdot |\mathcal{A}|} = \mathbb{R}^{nm}$ – in the latter case, we are considering a proper 1-to-1 mapping $\mathcal{S} \times \mathcal{A} \to \{1, 2, \dots, nm\}$.

Let us now fix a control policy $\pi : \mathcal{S} \to \mathcal{A}$, i.e., a mapping from states to actions. The stage cost of the policy π is denoted by $c^\pi \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n$ with elements $c^\pi(s) = c\big(s, \pi(s)\big)$ for $s \in \mathcal{S}$. The transition probability kernel of the resulting Markov chain under the policy π is denoted by $\mathbb{P}^\pi$, where $\mathbb{P}^\pi(s^+|s) = \mathbb{P}\big(s^+|s, \pi(s)\big)$ for $s, s^+ \in \mathcal{S}$. We also define the matrix $P^\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|} = \mathbb{R}^{n \times n}$, with elements $P^\pi(s, s^+) := \mathbb{P}^\pi(s^+|s)$ for $s, s^+ \in \mathcal{S}$, to be the corresponding transition probability *matrix*. The value of a policy is the expected, discounted, accumulative cost of following this policy

Domain	Optimization		Control	
Problem	Function $\widehat{f} : \mathbb{R}^\ell \times \Xi \to \mathbb{R}$, Random variable $\xi \sim \mathbb{P}$, $\min_x \{f(x) \coloneqq \mathbb{E}_{\mathbb{P}}[\widehat{f}(x,\xi)]\}$		State $s \in \mathcal{S}$, Control $a \in \mathcal{A}$, Dynamics $s^+ \sim \mathbb{P}(\cdot\|s,a)$, Cost $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $\min_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}_{\mathbb{P}}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, \pi(s_t)) \middle\| s_0 = s\right], \quad \forall s \in \mathcal{S}$	
Type	**Deterministic**	**Stochastic**	**Model-based**	**Model-free**
Equivalent characterization	$\nabla f(x^\star) = 0$	$\mathbb{E}_{\mathbb{P}}[\nabla \widehat{f}(x^\star, \xi)] = 0$	$v^\star = T(v^\star)$	$q^\star = \mathbb{E}_{\mathbb{P}}\left[\widehat{T}(q^\star, s^+)\right]$
Available oracle/info	$\nabla f(x)$ (Prob. distribution $\mathbb{P}$)	$\nabla \widehat{f}(x,\xi)$ (Samples $\hat{\xi}$)	$T(v)$ (Prob. kernel $\mathbb{P}$, Cost c)	$\widehat{T}(q, s^+)$ (Samples $(s, a, c(s,a), \hat{s}^+)$)
Transformation	x	$\longleftrightarrow$	v	
	∇f	$\longrightarrow$	$\operatorname{Id} - T$	
	$\operatorname{Id} - \nabla f$	$\longleftarrow$	T	
	$\nabla^2 f$	$\longrightarrow$	$I - \gamma P$	
	$I - \nabla^2 f$	$\longleftarrow$	γP	
		(x, ξ)	$\longleftrightarrow$	(q, s^+)
		$\nabla \widehat{f}$	$\longrightarrow$	$\operatorname{Id} - \widehat{T}$
		$\operatorname{Id} - \nabla \widehat{f}$	$\longleftarrow$	$\widehat{T}$
		$\nabla^2 \widehat{f}$	$\longrightarrow$	$I - \gamma \widehat{P}$
		$I - \nabla^2 \widehat{f}$	$\longleftarrow$	$\gamma \widehat{P}$

TABLE 1. Equivalence transformations: Id is the identity operator. T is the Bellman operator (4). $\widehat{T}$ is the sampled Bellman operator (6). The matrix $P = P(v)$ is the state transition probability matrix of the Markov chain under the greedy policy w.r.t. v. The matrix $\widehat{P} = \widehat{P}(q, \hat{s}^+)$ is the sampled state-action transition probability matrix of the Markov chain under the greedy policy w.r.t. q.

over an infinite-horizon trajectory: For the policy π, we define the value function $v^\pi \in \mathbb{R}^{|\mathcal{S}|} = \mathbb{R}^n$ with elements

$$v^\pi(s) \coloneqq \mathbb{E}_{s_{t+1} \sim \mathbb{P}^\pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t c^\pi(s_t) \,\middle|\, s_0 = s \right],$$

and the action-value function (a.k.a. Q-function) $q^\pi \in \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|} = \mathbb{R}^{nm}$ with elements

$$q^\pi(s,a) \coloneqq c(s,a) + \gamma \mathbb{E}_{s^+ \sim \mathbb{P}(\cdot|s,a)}[v^\pi(s^+)],$$

so that we also have $v^\pi(s) = q^\pi\big((s, \pi(s)\big)$ for each $s \in \mathcal{S}$. Given a value function v, let us also define $\pi_v : \mathcal{S} \to \mathcal{A}$ by

$$\pi_v(s) \in \operatorname*{argmin}_{a \in \mathcal{A}} \left\{ c(s,a) + \gamma \mathbb{E}_{s^+ \sim \mathbb{P}(\cdot|s,a)} \left[v(s^+) \right] \right\},$$

to be the *greedy policy w.r.t. v*. Similarly, for a Q-function q, define $\pi_q : \mathcal{S} \to \mathcal{A}$ by

$$\pi_q(s) \in \operatorname*{argmin}_{a \in \mathcal{A}} q(s,a),$$

to be the *greedy policy w.r.t. q*. The problem of interest is to control the MDP optimally, that is, to find the optimal policy π^* with the optimal (action-)value functions

$$v^\star = \min_\pi v^\pi, \quad q^\star = \min_\pi q^\pi, \tag{2}$$

so that the expected, discounted, infinite-horizon cost is minimized. Let us also note that the optimal policy, i.e., the minimizer of the preceding optimization problems, is the greedy policy w.r.t. $v^\star$ and $q^\star$, that is, $\pi^\star = \pi_{v^\star} = \pi_{q^\star}$.

Interestingly, the optimal (action-)value functions introduced in (2) can be equivalently characterized as the fixed point of two different operators each of which is useful depending on the available information (oracle):

(i) *Model-based* control: When we have access to the transition kernel and the cost function, the problem is usually characterized by the fixed-point problem $v^\star = T(v^\star)$, i.e.,

$$v^\star(s) = [T(v^\star)](s), \quad \forall s \in \mathcal{S}, \tag{3}$$

where $T : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is the *Bellman operator* given by

$$[T(v)](s) := \min_{a\in\mathcal{A}} \left\{ c(s,a) + \gamma \mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)} \left[v(s^+) \right] \right\}. \tag{4}$$

That is, the optimal value function $v^\star$ is the unique fixed-point of the Bellman operator T. The uniqueness follows from the fact that the operator T is a γ-contraction in ∞-norm. Observe that, in this case, T can be exactly computed given the model of the underlying MDP.

(ii) *Model-free* control: In real applications, the model is often not known, and instead one can generate samples. Examples of this are very large systems where identifying the model is prohibitively expensive but transitions between states can be observed and recorded, such as those in video games. This problem has been studied extensively in the reinforcement learning community and is often characterized as the *expected* fixed-point problem $q^\star = \mathbb{E}_{s^+\sim\mathbb{P}} \left[\widehat{T}(q^\star, s^+) \right]$, i.e.,

$$q^\star(s,a) = \mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)} \left[[\widehat{T}(q^\star, s^+)](s,a) \right], \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}, \tag{5}$$

where $\widehat{T} : \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|} \times \mathcal{S} \to \mathbb{R}^{|\mathcal{S}|\cdot|\mathcal{A}|}$ is the *sampled* Bellman operator given by[2]

$$[\widehat{T}(q, \hat{s}^+)](s,a) := c(s,a) + \gamma \min_{a^+\in\mathcal{A}} q(\hat{s}^+, a^+), \tag{6}$$

with $\hat{s}^+ \sim \mathbb{P}(\cdot|s,a)$ being a *sample* of the next state drawn from the distribution $\mathbb{P}(\cdot|s,a)$ for the pair (s,a).

2.2. **Optimization problem**

We now look at the root-finding characterization of the solution to convex optimization problems. Consider the minimization problem

$$\min_{x\in\mathbb{R}^\ell} \left\{ f(x) = \mathbb{E}_{\xi\sim\mathbb{P}}[\widehat{f}(x,\xi)] \right\}, \tag{7}$$

where the function $\widehat{f} : \mathbb{R}^\ell \times \Xi \to \mathbb{R}$ and the probability distribution $\mathbb{P}$ over Ξ are such that the function $f : \mathbb{R}^\ell \to \mathbb{R}$ is twice continuously differentiable and strongly convex. Much like the control problem, this problem can be considered in two settings:

[2]Strictly speaking, the provided sampled Bellman operator is the empirical version of the Bellman operator for the Q-function, given by $[T(q)](s,a) := c(s,a) + \gamma \mathbb{E}_{s^+\sim\mathbb{P}(\cdot|s,a)} \left[\min_{a^+\in\mathcal{A}} q(s^+, a^+) \right]$ for $(s,a) \in \mathcal{S}\times\mathcal{A}$.

(i) *Deterministic* optimization: Assuming that $\mathbb{P}$ in (7) is known and the corresponding expectation can be computed. Then, the unique minimizer $x^\star$ satisfies

$$\nabla f(x^\star) = 0. \tag{8}$$

(ii) *Stochastic* optimization: Now assume that $\mathbb{P}$ in (7) is unknown but can be sampled from. In this case, the minimizer $x^\star$ satisfies the expected root-finding problem

$$\mathbb{E}_{\xi\sim\mathbb{P}}[\nabla \widehat{f}(x^\star,\xi)] = 0, \tag{9}$$

where ∇ now denotes the partial derivative w.r.t. x. We note that, above, there is an underlying assumption that the differentiation w.r.t. x and expectation w.r.t. ξ can be operated in any order.

2.3. **Transformation**

Before providing the equivalence relations between optimization and control problems, let us provide an important result for the Bellman operator (see Appendix A.1 for the proof):

Lemma 2.1 (Jacobian of T)**.** *Let $\mathcal{S} = \{1,2,\dots,n\}$ and $\mathcal{A} = \{1,2,\dots,m\}$. If T is differentiable at v, then $\partial T(v) = \gamma P^{\pi_v}$, where π_v is the greedy policy w.r.t. v.*

Using the preceding result, we have $\partial(\mathrm{Id} - T)(v) = I - \gamma P(v)$, where $P(v) := P^{\pi_v}$, i.e., the state transition probability matrix of the greedy policy π_{v_k} w.r.t. v_k. Then, comparing the characterizations (3) and (8), we can draw the following equivalence relations between deterministic optimization and model-based control:

$$x \leftrightarrow v, \quad \left\{\begin{array}{rcl} \nabla f & \to & \mathrm{Id} - T \\ \mathrm{Id} - \nabla f & \leftarrow & T \end{array}\right., \quad \left\{\begin{array}{rcl} \nabla^2 f & \to & I - \partial T = I - \gamma P \\ I - \nabla^2 f & \leftarrow & \partial T = \gamma P \end{array}\right.,$$

where Id is the identity operator, I is the identity matrix, and $P = P(v)$ is the transition probability matrix of the Markov chain under the greedy policy w.r.t. v. Similarly, for stochastic optimization and model-free control, the characterizations (5) and (9) point to the following equivalence relations:

$$(x,\xi) \leftrightarrow (q,s^+), \quad \left\{\begin{array}{rcl} \nabla \widehat{f} & \to & \mathrm{Id} - \widehat{T} \\ \mathrm{Id} - \nabla \widehat{f} & \leftarrow & \widehat{T} \end{array}\right., \quad \left\{\begin{array}{rcl} \nabla^2 \widehat{f} & \to & I - \partial \widehat{T} = I - \gamma \widehat{P} \\ I - \nabla^2 \widehat{f} & \leftarrow & \partial \widehat{T} = \gamma \widehat{P} \end{array}\right.,$$

where $\widehat{P} = \widehat{P}(q,\hat{s}^+) \in \mathbb{R}^{nm\times nm}$ is the synchronously *sampled* transition probability matrix of the Markov chain under π_q with elements[3]

$$[\widehat{P}(q,\hat{s}^+)]\big((s,a),(s',a')\big) = \left\{\begin{array}{ll} 1, & s' = \hat{s}^+,\ a' = \pi_q(\hat{s}^+) \\ 0, & \text{otherwise} \end{array}\right., \quad (s,a),(s',a') \in \mathcal{S}\times\mathcal{A},$$

for each $(s,a),(s',a') \in \mathcal{S}\times\mathcal{A}$, where $\hat{s}^+ \sim \mathbb{P}(\cdot|s,a)$ is again a *sample* of the next state drawn from the distribution $\mathbb{P}(\cdot|s,a)$ for the state-action pair (s,a).

[3]Once again, strictly speaking, $\widehat{P}(q,\hat{s}^+)$ is the empirical version of the *state-action* transition probability matrix $P(q) \in \mathbb{R}^{nm\times nm}$ of the Markov chain under the greedy policy π_q w.r.t. q, with elements $[P(q)]\big((s,a),(s',a')\big) = \mathbb{P}(s'|s,a)$ if $a' = \pi_q(s')$ and $= 0$ otherwise, for $(s,a),(s',a') \in \mathcal{S}\times\mathcal{A}$.

3. Equivalent Algorithms

We now look at existing algorithms for optimization and control and their equivalence within the proposed framework. In particular, we show how the application of the proposed transformations on well-established optimization algorithms such as gradient descent, accelerated gradient descent, and Newton method leads to well-known control algorithms such as value iteration, accelerated value iteration, and policy iteration. We note that these equivalences have already been pointed out in the existing literature, however, mostly in a scattered way. An exception is [20] where the relation between *deterministic* optimization algorithms and *model-based* control algorithms are studied.

For tabular MDPs with $\mathcal{S} = \{1, \dots, n\}$ and $\mathcal{A} = \{1, \dots, m\}$, we have $v \in \mathbb{R}^n$ and $q \in \mathbb{R}^{nm}$ for the value function and the Q-function, respectively. Correspondingly, we have $T : \mathbb{R}^n \to \mathbb{R}^n$ with

$$T(v) = \textstyle\sum_{s \in \mathcal{S}} \, [T(v)](s) \cdot \mathbf{1}(s),$$

where $\mathbf{1}(s) \in \mathbb{R}^n$ is the unit vector for the state $s \in \mathcal{S}$, and $\widehat{T} : \mathbb{R}^{nm} \times \mathcal{S}^{nm} \to \mathbb{R}^{nm}$ with

$$\widehat{T}(q, \hat{s}^+) = \textstyle\sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \, [\widehat{T}(q, \hat{s}^+)](s, a) \; \mathbf{1}(s, a),$$

where $\mathbf{1}(s, a) \in \mathbb{R}^{nm}$ is the unit vector for the state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. Above, with some abuse of notation, $\hat{s}^+$ captures the dependence of the sampled Bellman operator $\widehat{T}$ on the specific samples $\hat{s}^+ \sim \mathbb{P}(\cdot|s, a)$, with one sample for each $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Moreover, to ease the exposition, we see any iterative algorithm as

$$y_{k+1} = y_k + d_k, \quad k = 0, 1, \dots$$

where $y_k = x_k$, v_k or q_k based on the context. In each setting, d_k represents the update vector between iterations k and $k + 1$. This form allows us to characterize algorithms in terms of d_k. A compact summary of this can be found in Table 2.

3.1. First-order methods

The celebrated gradient descent (GD) [31] method is characterized by $d_k = -\alpha_k \nabla f(x_k)$, where α_k is a properly chosen step-size. Applying the transformations of Table 1 on GD, we derive the so-called relaxed value iteration (VI) [30, 39, 19] with $d_k = -\alpha_k \big(v_k - T(v_k)\big)$, for model-based control. In particular, for the constant step-size $\alpha_k = 1$, we have the standard VI algorithm $v_{k+1} = T(v_k)$ [4]. The stochastic counterpart of GD (SGD) [42] is characterized by $d_k = -\alpha_k \nabla \widehat{f}(x_k, \hat{\xi}_k)$. Under the transformations of Table 1, SGD leads to the *synchronous* Q-learning (QL) algorithm [51, 26] with[4]

$$d_k = -\alpha_k \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big). \tag{10}$$

[4]This is the so-called *synchronous* update of the Q-function in *all* state-action pairs in each iteration, corresponding to the *parallel sampling model* introduced by [26].

Deterministic optimization $(y=x)$	**Model-based control** $(y=v)$	**Stochastic optimization** $(y=x)$	**Model-free control** $(y=q)$
$g(x) := \nabla f(x)$ $H(x) := \nabla^2 f(x)$	$g(v) := v - T(v)$ $H(v) := I - \gamma P(v)$	$\widehat{g}_k(x) := \nabla \widehat{f}(x, \hat{\xi}_k)$ $\widehat{H}_k(x) := \nabla^2 \widehat{f}(x, \hat{\xi}_k)$	$\widehat{g}_k(q) := q - \widehat{T}(q, \hat{s}_k^+)$ $\widehat{H}_k(q) := I - \gamma \widehat{P}(q, \hat{s}_k^+)$
GD [31]	Relaxed VI [4, 30]	SGD [42]	QL [51]
$d_k = -\alpha_k g(y_k)$		$d_k = -\alpha_k \widehat{g}_k(y_k)$	
Ployak GD [38]	Momentum VI [19]	Momentum SGD [53]	Speedy QL [17], NeSA [12], Momentum QL [52]
$d_k = -\alpha_k g(y_k) + \beta_k d_{k-1}$		$\begin{cases} d'_{k-1} = \widehat{g}_k(y_{k-1}) - \widehat{g}_k(y_k) \\ d_k = -\alpha_k \widehat{g}_k(y_k) + \beta_k d'_{k-1} + \delta_k d_{k-1} \end{cases}$	
Nesterov GD [37]	Accelerated VI [19]		
$d_k = -\alpha_k g(y_k + \beta_k d_{k-1}) + \beta_k d_{k-1}$			
NM	PI [23]	SNR [43, 13]	Zap QL [13]
$d_k = -\left[H(y_k)\right]^{-1} g(y_k)$		$\begin{cases} D_k = (1-\beta_k) D_{k-1} + \beta_k \widehat{H}_k(y_k) \\ d_k = -\alpha_k D_k^{-1} \widehat{g}_k(y_k) \end{cases}$	

Table 2. Equivalent algorithms: d_k is the update vector as in a generic iterative scheme $y_{k+1} = y_k + d_k$ for $k = 0, 1 \dots$. $\alpha_k, \beta_k, \delta_k > 0$ are properly chosen step sizes. The second row contains definitions of the mathematical objects used in the rows below. All the provided model-free control algorithms are synchronous, i.e., all the state-action pairs in the Q-function are updated at each iteration. (S)GD: (stochastic) gradient descent; VI: value iteration; NM: Newton method; PI: policy iteration; QL: Q-learning; SNR: stochastic Newton-Raphson.

3.2. Accelerated methods

In the so-called momentum-based algorithms, the update vector d_k is specified by gradient oracles but also depends on d_{k-1}. One such algorithm is GD with Polyak momentum (Polyak GD) [38], a.k.a. heavy ball method, characterized by $d_k = -\alpha_k \nabla f(x_k) + \beta_k d_{k-1}$. Another well-known momentum-based algorithm is GD with Nesterov acceleration (Nesterov GD) [37] with update vector $d_k = -\alpha_k \nabla f(x_k + \beta_k d_{k-1}) + \beta_k d_{k-1}$. With a proper choice of the step-sizes α_k and β_k, these schemes can be shown to accelerate the convergence rate, compared to the standard GD, for particular classes of objective functions [38, 36]. The corresponding model-based control algorithms, using the transformations of Table 1, are *momentum VI* [19] with $d_k = -\alpha_k\big(v_k - T(v_k)\big) + \beta_k d_{k-1}$, and *accelerated VI* [19] with $d_k = -\alpha_k\big(v_k + \beta_k d_{k-1} - T(v_k + \beta_k d_{k-1})\big) + \beta_k d_{k-1}$. However, the convergence of the preceding accelerated schemes is in general not guaranteed. In [19], the authors address this issue by *safeguarding*, i.e., combining the accelerated VI with the standard VI. For accelerating SGD, a direct combination of Polyak momentum or Nesterov acceleration with SGD has been shown to lead to no better (and even worse) performance in terms of convergence rate [53, 27]. At least, when it comes to almost sure convergence, [33] reports the same rate of convergence for SGD with Polyak momentum and SGD with Nesterov acceleration as for standard SGD. Nevertheless, modifications of momentum-based acceleration methods have led to a range of accelerated SGD algorithms with faster convergence rates with specific assumptions on the problem data [27, 32, 1].

The idea of using momentum for accelerating QL has also attracted some interest. In particular, applying the transformations of Table 1 on a generic momentum SGD [53] with

$$\begin{cases} d'_{k-1} = \nabla \widehat{f}(x_{k-1}, \hat{\xi}_k) - \nabla \widehat{f}(x_k, \hat{\xi}_k), \\ d_k = -\alpha_k \nabla \widehat{f}(x_k, \hat{\xi}_k) + \beta_k d'_{k-1} + \delta_k d_{k-1}, \end{cases}$$

and step-sizes $\alpha_k, \beta_k, \delta_k > 0$, we obtain the *speedy QL* [17], *NeSA* [12], and *momentum QL* [52] algorithms with

$$\begin{cases} d'_{k-1} = \big(q_{k-1} - \widehat{T}(q_{k-1}, \hat{s}_k^+)\big) - \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big), \\ d_k = -\alpha_k \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big) + \beta_k d'_{k-1} + \delta_k d_{k-1}. \end{cases} \tag{11}$$

The difference between these three algorithms is in the choice of the step-sizes $\alpha_k, \beta_k, \delta_k > 0$.

3.3. Second-order methods

In second-order algorithms, d_k is specified by both the gradient and the Hessian oracles. The *damped* Newton method is one such algorithm with $d_k = -\alpha_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$. The *pure* Newton step with $\alpha_k = 1$, has a *local* quadratic convergence, if in addition to f being strongly convex, the Hessian is Lipschitz-continuous [10, Thm. 5.3]. Globally, however, the pure Newton method can lead to divergence. This is the reason behind introducing the step-size $\alpha_k < 1$ in the damped version which can be used to guarantee a global linear convergence. We can use the transformations of Table 1 in order to transform the Newton method into a model-based control with $d_k = -\big(I - \gamma P(v_k)\big)^{-1}(v_k - T(v_k))$, where $P(v_k)$ is the transition probability matrix of the Markov chain under the greedy policy w.r.t. v_k. The derived model-based control algorithm then corresponds to the well-known policy iteration (PI) algorithm [41] with $v_{k+1} = \big(I - \gamma P(v_k)\big)^{-1} c^{\pi_{v_k}}$, where $c^{\pi_{v_k}}$ is the vector of stage costs corresponding to the greedy policy π_{v_k} w.r.t. v_k. Indeed, the PI algorithm is equivalent to the semi-smooth Newton method with a local quadratic convergence rate [15]. The second-order scheme has also been combined with the VI algorithm by using the *smooth* Bellman operation in which the maximization operation is approximated by a differentiable function, e.g., log-sum-exp [44]. This idea has been recently used to propose the generalized second-order VI with a quadratic convergence rate [25].

In the model-free case, the stochastic version of the Newton method [43] has been a source of inspiration for developing second-order-type Q-learning algorithms. In particular, the stochastic Newton-Raphson (SNR) [43] algorithm with

$$\begin{cases} D_k = (1 - \beta_k) D_{k-1} + \beta_k \nabla^2 \widehat{f}(x_k, \hat{\xi}_k), \\ d_k = -\alpha_k D_k^{-1} \nabla \widehat{f}(x_k, \hat{\xi}_k), \end{cases}$$

was used for developing the *Zap QL* algorithm [13] with

$$\begin{cases} D_k = (1 - \beta_k)\ D_{k-1} + \beta_k\ \mathbf{1}(s_k, a_k)\ \big(\mathbf{1}(s_k, a_k) - \gamma\ \mathbf{1}\big(\hat{s}_k^+, \pi_k(\hat{s}_k^+)\big)\big)^\top, \\ d_k = -\alpha_k\ D_k^{-1}\ \big(q_k(s_k, a_k) - [\widehat{T}(q_k, \hat{s}_k^+)](s_k, a_k)\big)\ \mathbf{1}(s_k, a_k), \end{cases}$$

where $\pi_k(\hat{s}_k^+) = \operatorname{argmin}_{a \in \mathcal{A}} q_k(\hat{s}_k^+, a)$ is the greedy action w.r.t. q_k evaluated at the sampled next sate $\hat{s}_k^+ \sim \mathbb{P}(\cdot | s_k, a_k)$. Note that the preceding algorithm involves updating one entry of the action-value function q_k at each iteration k, corresponding to the state-action pair (s_k, a_k) chosen at iteration k

– recall that $\mathbf{1}(s,a) \in \mathbb{R}^{nm}$ is the unit vector corresponding to the state-action pair (s,a). The implementation of Zap QL algorithm with *synchronous* update of the Q-function in all state-action pairs in each iteration is then characterized by

$$\begin{cases} D_k = (1-\beta_k) D_{k-1} + \beta_k \big(I - \gamma \widehat{P}(q_k, \hat{s}_k^+)\big), \\ d_k = -\alpha_k D_k^{-1} \big(q_k - \widehat{T}(q_k, \hat{s}_k^+)\big), \end{cases} \tag{12}$$

where $\widehat{P}(q,\hat{s}^+)$ is the synchronously sampled state-action transition probability matrix of the Markov chain under the greedy policy w.r.t. q. Note that (12) is exactly the SNR algorithm under the transformations of Table 1.

4. Quasi-Policy Iteration (QPI)

While Newton method (NM) has a higher convergence rate compared to gradient descent (GD), it suffers from a higher per-iteration computational cost. To be precise, consider again the unconstrained minimization problem $\min_{x\in\mathbb{R}^\ell} f(x)$, where f is twice continuously differentiable and strongly convex with a Lipschitz-continuous Hessian. Then, the GD update rule $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, with a proper choice of step-size α_k, converges *linearly* [10, Thm. 3.12] with $O(\ell)$ per-iteration complexity (disregarding the complexity of gradient oracle). On the other hand, the NM update rule $x_{k+1} = x_k - \alpha_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$, with a proper choice of step-size α_k, has a local *quadratic* convergence rate [10, Thm. 5.3] with $O(\ell^3)$ per-iteration complexity, assuming direct inversion (and disregarding the complexity of gradient and Hessian oracles).

Quasi-Newton methods (QNMs) are a class of methods that allow for a trade-off between computational complexity and (local) convergence rate. To do so, these methods use a Newton-type update rule

$$x_{k+1} = x_k - \alpha_k \widetilde{H}_k^{-1} \nabla f(x_k),$$

where $\widetilde{H}_k$ is an *approximation* of the true Hessian $\nabla^2 f(x_k)$ at iteration k. Different QNMs use different approximations of the Hessian with A generic approximation scheme in QNMs is

$$\widetilde{H}_k = \operatorname*{argmin}_{H\in\mathbb{R}^{\ell\times\ell}} \left\{ \|H - H_{\text{prior}}\|_F^2 \; : \; Hr_i = b_i, \; i = 1,\dots,j \right\}, \tag{13}$$

which minimizes the distance (in Frobenius norm) to a given prior H_{prior} subject to j (≥ 1) linear constraints specified by $r_i, b_i \in \mathbb{R}^\ell$. This leads to the approximation $\widetilde{H}_k$ being a rank-j update of the prior H_{prior}, i.e.,

$$\widetilde{H}_k = H_{\text{prior}} + (B - H_{\text{prior}} R)(R^\top R)^{-1} R^\top,$$

where $R = (r_1,\dots,r_j)$, $B = (b_1,\dots,b_j) \in \mathbb{R}^{\ell\times j}$. Hence, $\widetilde{H}_k^{-1}$ can be easily computed based on H_{prior}^{-1} using the Woodbury formula. Different choices of the prior and the linear constraints in the generic approximation scheme above lead to different QNMs. For example, by choosing the so-called *secant conditions* with $r_i = x_{k-i+1} - x_{k-i}$ and $b_i = \nabla f(x_{k-i+1}) - \nabla f(x_{k-i})$ as linear constraints and $H_{\text{prior}} = I$ as the prior, we derive Anderson mixing with memory j [2].

In this section, following a similar idea, we propose the *quasi-policy iteration (QPI)* algorithm by incorporating an efficiently computable approximation of the "Hessian" $H = I - \gamma P$ in the PI algorithm. We note that the authors in [54] and [16] also propose the combination of Anderson mixing with optimal control algorithms. However, the QPI algorithm is fundamentally different in the sense that it approximates the transition probability matrix P using a different set of constraints that are specific to the optimal control algorithms.

4.1. QPI Algorithm

For $k \in \{0, 1, 2, \ldots\}$, let

$$c_k := c^{\pi_{v_k}}, \quad P_k := P^{\pi_{v_k}}, \quad T_k := T(v_k).$$

(Recall that $c^{\pi_{v_k}}$ and $P^{\pi_{v_k}}$ are the stage cost and the state transition probability matrix of the greedy policy π_{v_k} w.r.t. v_k, respectively.) Recall the PI update rule

$$v_{k+1} = (I - \gamma P_k)^{-1} c_k = v_k - (I - \gamma P_k)^{-1}(v_k - T_k).$$

Inspired by the QNM approximation scheme (13), we propose the generic QPI update rule

$$v_{k+1} = v_k - (I - \gamma \widetilde{P}_k)^{-1}(v_k - T_k), \tag{14}$$

where

$$\widetilde{P}_k = \operatorname*{argmin}_{P \in \mathbb{R}^{n \times n}} \left\{ \|P - P_{\text{prior}}\|_F^2 \ : \ Pr_i = b_i, \ i = 1, \ldots, j \right\}. \tag{15}$$

Observe that instead of approximating the complete Hessian $H_k := (I - \gamma P_k)^{-1}$ similar to standard QNMs, we are only approximating P_k. This choice particularly allows us to exploit the problem structure in order to form novel prior and constraints as we discuss next.

Regarding the constraints, the problem structure gives us two linear equality constraints: First, P_k is a row stochastic matrix, i.e.,

$$P_k \mathbf{1} = \mathbf{1}, \tag{16}$$

and hence we can set $r_1 = b_1 = \mathbf{1}$. Second, we can use the fact that the Bellman operator T is locally affine. In particular, from the definition (4) of the Bellman operator, it follows that $T(v) = c^{\pi_v} + \gamma P^{\pi_v} v$. Thus,

$$T_k = c_k + \gamma P_k v_k \ \Rightarrow \ P_k v_k = \gamma^{-1}(T_k - c_k), \tag{17}$$

and we can set $r_2 = v_k$ and $b_2 = \gamma^{-1}(T_k - c_k)$. Note that, unlike the standard secant conditions in QNMs, the constraints (16) and (17) hold *exactly*. Next to be addressed is the choice of the prior. For the proposed QPI algorithm, we consider a generic and computationally advantageous choice for the prior, namely, the uniform distribution: $P_{\text{prior}} = \frac{1}{n}E = \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. Incorporating the constraints (16) and (17) with the uniform prior, we propose the approximation

$$\widetilde{P}_k = \operatorname*{argmin}_{P \in \mathbb{R}^{n \times n}} \left\{ \|P - \frac{1}{n}E\|_F^2 \ : \ P\mathbf{1} = \mathbf{1}, \ Pv_k = \gamma^{-1}(T_k - c_k) \right\}. \tag{18}$$

The update rule (14) using the approximation (18) is however not necessarily a contraction. The same problem also arises in similar algorithms such as Anderson accelerated VI [54] and Nesterov accelerated VI [19]. Here, we follow the standard solution for this problem, that is, safeguarding the QPI update against the standard VI update based on the Bellman error

$$\theta_k := \|v_k - T_k\|_\infty .$$

To be precise, at each iteration $k = 0, 1, \ldots$, we consider the *safeguarded* QPI update rule as follows

$$\begin{array}{ll} \text{(QPI)} & \text{compute } v_{k+1} \text{ according to (14), (18);} \\ \text{(Safeguard)} & \textbf{if } \theta_{k+1} > \gamma^{k+1}\theta_0, \textbf{ then } v_{k+1} = T_k. \end{array} \tag{19}$$

Remark 4.1 (Other priors/constraints)**.** *We can also consider other priors and/or add extra constraints to the minimization problem* (18) *to impose a particular structure on the approximate transition probability matrix $\widetilde{P}_k$. For instance, a natural constraint is to require this matrix to be entry-wise non-negative so that $\widetilde{P}_k$ is indeed a probability transition matrix; or, one can consider a given sparsity pattern on P_k by choosing a proper prior P_{prior}. However, incorporating this extra information may lead to the corresponding minimization problem not having a closed-form and/or low-rank solution, and hence undermining the computational efficiency of the proposed algorithm. In this regard, we note that the minimization problem* (18) *has a closed-form solution of rank at most two; see the proof of Theorem 4.2 in Appendix A.2 for details.*

The following theorem summarizes the discussion above by providing the QPI update rule explicitly (see Appendix A.2 for the proof).

Theorem 4.2 (QPI convergence & complexity)**.** *The update rule* (14) *using the approximation* (18) *admits the closed-form solution*

$$v_{k+1} = (1-\delta_k)T_k + \delta_k c_k + \lambda_k \mathbf{1}, \tag{20a}$$

where the scalar coefficients are given by

$$\begin{aligned} &z_k = c_k - \tfrac{\mathbf{1}^\top c_k}{n}\mathbf{1} \in \mathbb{R}^n, \quad g_k = v_k - T_k \in \mathbb{R}^n, \quad y_k = g_k - \tfrac{\mathbf{1}^\top g_k}{n}\mathbf{1} \in \mathbb{R}^n, \\ &\delta_k = \begin{cases} 0, & v_k^\top(y_k + z_k) = 0 \\ \frac{v_k^\top y_k}{v_k^\top(y_k+z_k)}, & \textit{otherwise} \end{cases} \in \mathbb{R}, \quad \lambda_k = \tfrac{\gamma}{n(1-\gamma)}\mathbf{1}^\top\big((\delta_k - 1)g_k + \delta_k c_k\big) \in \mathbb{R}. \end{aligned} \tag{20b}$$

Moreover, each iteration of the QPI update rule (20) *with the safeguarding* (19) *is a γ-contraction in the infinity norm and has a time complexity of $\mathcal{O}(n^2 m)$.*

Some remarks are in order regarding the preceding result. First, observe that the QPI update rule (20a) is a modification of the standard VI update rule $v_{k+1} = T(v_k)$ using two new vectors, namely, $c_k - T(v_k)$ and the all-one vector $\mathbf{1}$, with adaptive coefficients δ_k and λ_k, respectively. These new directions are the direct product of the prior and the linear equality constraints used in the approximation (18).

Second, the safeguarded QPI update rule has the same per-iteration complexity as VI , i.e., $\mathcal{O}(n^2 m)$. Moreover, the convergence of QPI is ensured via safeguarding with VI, which leads to the same theoretically guaranteed linear convergence with rate γ as for VI. However, as we will show in the numerical examples below, we observe an empirically faster convergence for QPI with its rate showing less sensitivity to γ similar to PI. The pseudo-code for the safeguarded QPI algorithm is provided in Algorithm 1 in Appendix B.

4.2. Extension to model-free control: QPL algorithm

We now introduce the quasi-policy learning (QPL) algorithm as the extension of QPI for model-free control problems with access to samples through a generative model. The basic idea is to implement the stochastic version of the QPI update rule proposed above (for the Q-function) using the samples. For that, we use the *sampled* Bellman operator $\widehat{T}(\cdot, \hat{s}_k^+)$, evaluated at the sampled next states $\hat{s}_k^+$ at iteration k, as a surrogate for the Bellman operator $T(\cdot)$. To be precise, let

$$\widehat{T}_k := \widehat{T}(q_k, \hat{s}_k^+),$$

at each iteration k. Also, let $c \in \mathbb{R}^{nm}$ be the vector of stage cost (with the same state-action ordering as the Q-function $q_k \in \mathbb{R}^{nm}$). We note that since the proposed QPL algorithm is synchronous with one sample for each state-action pair in each iteration, we have access to the complete stage cost c after the first iteration and can treat it as an input to the algorithm without loss of generality. The update rule of the model-free QPL algorithm is then as follows:

$$q_{k+1} = (1-\alpha_k) q_k + \alpha_k \big((1-\delta_k)\widehat{T}_k + \delta_k c + \lambda_k \mathbf{1}\big), \tag{21a}$$

where

$$\begin{aligned} &z = c - \tfrac{\mathbf{1}^\top c}{nm}\mathbf{1} \in \mathbb{R}^{nm}, \quad \widehat{g}_k = q_k - \widehat{T}_k \in \mathbb{R}^{nm}, \quad y_k = \widehat{g}_k - \tfrac{\mathbf{1}^\top \widehat{g}_k}{nm}\mathbf{1} \in \mathbb{R}^{nm}, \\ &\delta_k = \begin{cases} 0, & q_k^\top (y_k + z) = 0 \\ \frac{q_k^\top y_k}{q_k^\top (y_k+z)}, & \text{otherwise} \end{cases} \in \mathbb{R}, \quad \lambda_k = \tfrac{\gamma}{nm(1-\gamma)} \mathbf{1}^\top \big((\delta_k - 1)\widehat{g}_k + \delta_k c\big) \in \mathbb{R}. \end{aligned} \tag{21b}$$

and α_k is the diminishing learning rate of the algorithm, e.g., $\alpha_k = 1/(k+1)$. Compared to the standard Q-learning (QL) update rule, i.e., $q_{k+1} = (1-\alpha_k)q_k + \alpha_k \widehat{T}_k$, QPL uses the two additional vectors $c - \widehat{T}_k$ and the all-one vector $\mathbf{1}$ with adaptive coefficients in its update rule.

Similar to the model-based case, the proposed QPL update rule is not necessarily convergent. To address this issue, we again use the basic idea of safeguarding. However, in this case, we safeguard the QPL update against the standard QL update based on the *sampled* Bellman error

$$\widehat{\theta}_k := \|q_k - \widehat{T}_k\|_\infty .$$

To be precise, we run a QL algorithm

$$q_{k+1}^{\mathrm{QL}} = (1-\alpha_k) q_k^{\mathrm{QL}} + \alpha_k \widehat{T}_k^{\mathrm{QL}},$$

in parallel with the QPL algorithm using the same initialization $q_0^{\mathrm{QL}} = q_0$ and the same samples for computing the corresponding sampled Bellman operator and error

$$\widehat{T}_k^{\mathrm{QL}} := \widehat{T}(q_k^{\mathrm{QL}}, \hat{s}_k^+), \quad \widehat{\theta}_k^{\mathrm{QL}} := \|q_k^{\mathrm{QL}} - \widehat{T}_k^{\mathrm{QL}}\|_\infty .$$

We then use the sampled Bellman error of QL to *safeguard* the QPL update rule as follows

$$\begin{aligned} &\text{(QPL)} && \text{compute } q_{k+1} \text{ according to (21);} \\ &\text{(Safeguard)} && \textbf{if } k > K_{\mathrm{sg}} \text{ and } \textstyle\sum_{i=0}^{k+1} \widehat{\theta}_i > \sum_{i=0}^{k+1} \widehat{\theta}_i^{\mathrm{QL}}, \textbf{ then } q_{k+1} = (1-\alpha_k) q_k + \alpha_k \widehat{T}_k \ . \end{aligned} \tag{22}$$

In particular, in order to increase the robustness against the stochasticity of the samples, the safeguard is activated (i) after K_{sg} iterations and (ii) based on the *accumulated* sampled Bellman errors over the entire history of iterations. The following theorem summarizes properties of the proposed QPL algorithm (see Appendix A.3 for the proof).

Theorem 4.3 (QPL convergence & complexity)**.** *The safeguarded QPL algorithm* (22) *has a per-iteration complexity of* $\mathcal{O}(nm^2)$ *and converges with at least the same rate as QL.*

Regarding the preceding result, we note that the per-iteration time complexity of QPL *with safeguard* is the same as that of the (synchronous) QL algorithm. Moreover, while the safeguard guarantees convergence with the same rate as QL, QPL has an empirical performance similar to that of Zap QL as shown in the numerical experiments below. Algorithm 2 in Appendinx B provides the pseudo-code for the safeguarded QPL algorithm.

5. Numerical simulations

We now compare the performance of the proposed algorithms in this study with that of the standard existing algorithms in solving 50 instances of the optimal control problems of randomly generated Garnet MDPs [3] with $n = 50$ states and $m = 5$ actions. A detailed description of these MDPs are provided in Appendix C.1.

In the model-based case, the proposed QPI algorithm is compared with the following algorithms:

- VI (value iteration);
- NVI (VI with Nesterov acceleration) [19];
- AVI (VI with Anderson acceleration) [16];
- PI (policy iteration).

For AVI, we use a memory of one leading to a rank-one update (of the identity matrix) for approximating the Hessian so that it is comparable with the rank-one update of the uniform distribution for approximating the transition probability matrix in QPI. See Appendices C.2 and C.3 for the exact update rules of NVI and AVI, respectively. We note that since NVI and AVI are not guaranteed to converge, we safeguard them using VI (using the same safeguarding rule (19) used for QPI). All the model-based algorithms are intialized by $v_0 = \mathbf{0}_n$ with termination condition $\|v_k - T(v_k)\|_\infty \leq \epsilon = 10^{-6}$.

In the model-free case, the proposed QPL algorithm is compared with the following algorithms:

- QL (Q-learning) as in (10) with $\alpha_k = \frac{1}{(k+1)}$;
- SQL (speedy QL) as in (11) with $\alpha_k = \frac{1}{(k+1)}$ and $\beta_k = \delta_k = 1 - \frac{2}{(k+1)}$ [17];
- ZQL (Zap QL) as in (12) with $\alpha_k = \beta_k = \frac{1}{(k+1)}$ [13].

All the model-free algorithms are initialized by $q_0 = \mathbf{0}_{nm}$ and terminated after $K = 10^4$ iterations with a synchronous sampling of all state-action pairs at each iteration. For QPL, the safeguard is activated after $K_{\text{sg}} = 100$ iterations. For each model-free algorithm, we report the *average* of the normalized error $\|q_k - q^\star\|_\infty / \|q^\star\|_\infty$ over the 50 runs of the algorithm.

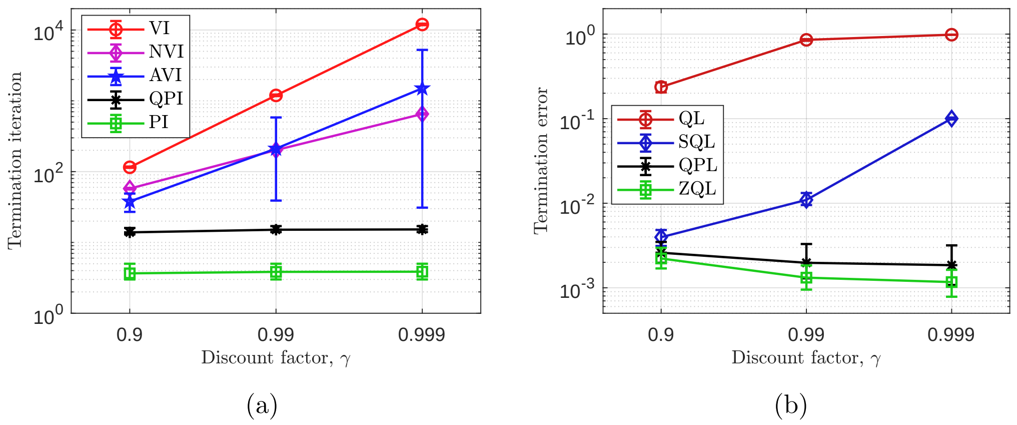
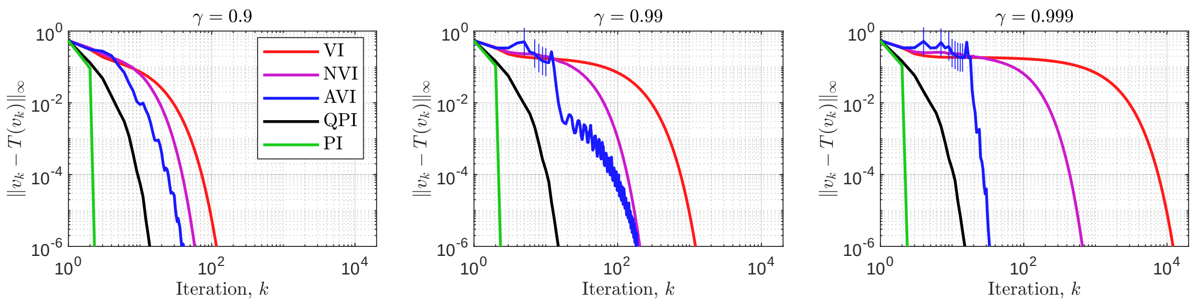


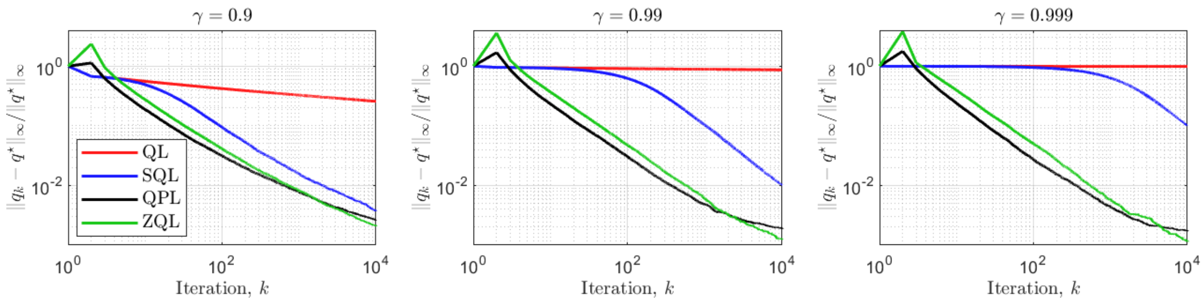
FIGURE 1. Average performance over 50 instances of the control problem for three different values of γ. The bars show the minimum and maximum performances: (a) model-based algorithms - the reported performance is the iteration number k at which the algorithm terminates according to the condition $\|v_k - T(v_k)\|_\infty \le \epsilon = 10^{-6}$; (b) model-free algorithms – the reported performance is the error $\|q_K - q^\star\|_\infty \,/\, \|q^\star\|_\infty$ for $K = 10^4$ (averaged over 50 runs).

The average performance of the algorithms over the considered 50 instances of the control problem for increasing values of the discount factor γ is depicted in Figure 1. As an illustration, the result of the simulations for a single stance of the optimal control problem is shown in Figure 2.

We begin with discussing the results of our numerical simulations for the model-based algorithms. As shown in Figure 1a, VI and NVI show a linear convergence with a rate depending on γ. In particular, as we increase γ from 0.9 to 0.999, we observe more than a tenfold increase in the number of iterations required for VI and NVI to terminate. Indeed, these algorithms only use first-order information, and their rate of convergence is determined by γ. On the other hand, PI converges with a quadratic rate in 3 to 5 iterations, independent of γ. Now, observe that for the considered class of the randomly generated Garnet MDPs, the proposed (model-based) QPI algorithm is showing a similar behavior as PI, terminating in less than 20 iterations, independent of γ. In this regard, recall that the per-iteration computational complexity of QPI is of the same order as VI. Moreover, comparing the performance of QPI with AVI (its counterpart in the class of quasi-Newton methods), we also see the importance of newly introduced linear constraints and prior for approximation of transition probability matrix as in (18). In particular, for larger values of γ, there is a lot of variance in the performance of AVI, with the average behavior being worse than NVI. Moreover, a closer look to Figure 2a shows a very poor performance of AVI for $\gamma = 0.99$ and $\gamma = 0.999$ in the first few iterations leading to multiple instances of safeguard activation, while for QPI the safeguard is never activated. We note that this behavior was observed in all of the considered instances of the optimal control problem, that is, the safeguard was never activated for QPI in any of the instances of the Garnet MDP with the considered three values of the discount factor (the corresponding results are not reported here).



(a)



(b)

FIGURE 2. Performance in a *single* instance of the control problem for three values of γ: (a) model-based algorithms; (b) model-free algorithms with the reported error being the average over 50 runs. The bars indicate the iterations at which the safeguard is activated (for NVI, AVI, QPI, and QPL).

A similar behavior is seen in the model-free case as shown in Figure 1b. As can be seen, the performance of QL and SQL (the first-order methods) deteriorates as γ increases. However, ZQL (the second-order method that estimates the transition probability matrix by averaging over the samples) leads to almost the same level of error after a fixed number of iterations for different values of γ. Now, observe that the proposed (model-free) QPL algorithm has a similar performance as ZQL with its rate of convergence being less sensitive to γ. We also note that in the performed numerical simulations, the safeguard of QPL was never activated in any of the runs for the considered instances of the optimal control problem. This can be seen particularly for the problem instance of Figure 2b.

Appendix A. Technical Proofs

A.1. Proof of Lemma 2.1

Let us define the matrix $P^a \in \mathbb{R}^{n\times n}$ with entries $P^a(s,s^+) = \mathbb{P}\left(s^+|s,a\right)$, for every control action $a \in \mathcal{A}$. Fix $s \in \mathcal{S}$, and observe that

$$\partial_v\big([T(v)](s)\big) = \partial_v\left(\min_{a\in\mathcal{A}}\left\{c(s,a) + \gamma\mathbb{E}_{\mathbb{P}(\cdot|s,a)}[v(s^+)]\right\}\right) = \partial_v\left(\min_{a\in\mathcal{A}}\left\{c(s,a) + \gamma\cdot P^a(s,\cdot)\cdot v\right\}\right)$$

where $P^a(s,\cdot)$ is the s-th row of P^a. Then, using the envelope theorem [45], we have

$$\partial_v\big([T(v)](s)\big) = \partial_v\Big(c\big(s,\pi_v(s)\big) + \gamma \cdot P^{\pi_v}(s,\cdot)\cdot v\Big) = \gamma \cdot P^{\pi_v}(s,\cdot).$$

Hence, $\partial T(v) = \gamma P^{\pi_v}$.

A.2. **Proof of Theorem 4.2**

First, let us show that the two equality constraints in the minimization problem (18) are linearly dependent if and only if $v_k^\top(y_k+z_k)=0$. In this regard, observe that the two equality constraints are linearly dependent if and only if $v_k = \rho\mathbf{1}$ for any $\rho\in\mathbb{R}$: For $\rho=0$, the second equality constraint becomes trivial; and, for $\rho\neq 0$, the two constraints become equivalent. On the other hand, we have

$$\begin{aligned}
v_k^\top(y_k+z_k) &= v_k^\top\big(g_k + c_k - \frac{\mathbf{1}^\top(g_k+c_k)}{n}\mathbf{1}\big) = v_k^\top\big(v_k - T_k + c_k - \frac{\mathbf{1}^\top(v_k - T_k + c_k)}{n}\mathbf{1}\big)\\
&= v_k^\top(I - \frac{1}{n}E)(I-\gamma P_k)v_k,
\end{aligned}$$

where, for the last equality, we used $T_k = c_k + \gamma P_k v_k$. Then, since $(I-\gamma P_k)$ is non-singular and $I - \frac{1}{n}E$ has only one zero eigenvalue corresponding to the eigenvector $\mathbf{1}$, we have

$$\begin{aligned}
v_k^\top(y_k+z_k) = 0 &\Leftrightarrow (I-\gamma P_k)v_k = \bar\rho\mathbf{1} \Leftrightarrow v_k = \bar\rho(I-\gamma P_k)^{-1}\mathbf{1} \Leftrightarrow v_k = \frac{\bar\rho}{1-\gamma}\mathbf{1} && (\bar\rho\in\mathbb{R})\\
&\Leftrightarrow v_k = \rho\mathbf{1}, && (\rho\in\mathbb{R})
\end{aligned}$$

where we used the fact that $(I-\gamma P_k)\mathbf{1} = (1-\gamma)\mathbf{1}$. Hence, the constraints in (18) are linearly dependent if and only if $v_k^\top(y_k+z_k)=0$.

We now consider the update rule (20) for the case $v_k^\top(y_k+z_k)\neq 0$. Define $R \coloneqq (\mathbf{1}, v_k)$, $B \coloneqq \big(\mathbf{1}, \gamma^{-1}(T_k - c_k)\big)\in\mathbb{R}^{n\times 2}$ so that the minimization problem (18) can be written as

$$\widetilde{P}_k = \operatorname*{argmin}_{P\in\mathbb{R}^{n\times n}}\Big\{\|P - \frac{1}{n}E\|_F^2 \;:\; PR = B\Big\}.$$

Note that, since $v_k^\top(y_k+z_k)\neq 0$ and hence v_k and $\mathbf{1}$ are linearly independent, R is of full column rank. The solution to the preceding problem is given by

$$\widetilde{P}_k = \frac{1}{n}E + (B - \frac{1}{n}ER)(R^\top R)^{-1}R^\top.$$

Now, observe that

$$\begin{aligned}
(B - \frac{1}{n}ER) &= \begin{bmatrix}\mathbf{1} & \frac{1}{\gamma}(T_k - c_k)\end{bmatrix} - \frac{1}{n}E\begin{bmatrix}\mathbf{1} & v_k\end{bmatrix} = \begin{bmatrix}\mathbf{1} - \frac{1}{n}E\mathbf{1} & \frac{1}{\gamma}(T_k - c_k) - \frac{1}{n}Ev_k\end{bmatrix}\\
&= \begin{bmatrix}\mathbf{0}_n & \frac{1}{\gamma}(T_k - c_k) - \frac{\mathbf{1}^\top v_k}{n}\mathbf{1}\end{bmatrix},
\end{aligned}$$

and

$$(R^\top R)^{-1} = \left(\begin{bmatrix}\mathbf{1}^\top\\ v_k^\top\end{bmatrix}\begin{bmatrix}\mathbf{1} & v_k\end{bmatrix}\right)^{-1} = \begin{bmatrix} n & v_k^\top\mathbf{1}\\ v_k^\top\mathbf{1} & v_k^\top v_k\end{bmatrix}^{-1} = \frac{1}{\alpha}\begin{bmatrix} v_k^\top v_k & -\mathbf{1}^\top v_k\\ -\mathbf{1}^\top v_k & n\end{bmatrix},$$

where $\alpha := nv_k^\top v_k - (\mathbf{1}^\top v_k)^2$. Therefore, we have $\widetilde{P}_k = \frac{1}{n}E + \alpha^{-1}uw^\top$, where

$$u := \frac{1}{\gamma}\big(T_k - c_k - \frac{\gamma \mathbf{1}^\top v_k}{n}\mathbf{1}\big),\ w := n\big(I - \frac{1}{n}E\big)v_k.$$

That is, the approximation $\widetilde{P}_k$ is a rank-two matrix. Next, using the Woodbury formula and the fact that $(I - \frac{\gamma}{n}E)^{-1} = I + \beta E$ with $\beta = \frac{\gamma}{n(1-\gamma)}$, we have

$$\begin{aligned}(I - \gamma\widetilde{P}_k)^{-1} &= (I - \frac{\gamma}{n}E - \gamma u\alpha^{-1}w^\top)^{-1}\\ &= \Big(I + \gamma(I+\beta E)u\big(\alpha - \gamma w^\top(I+\beta E)u\big)^{-1}w^\top\Big)(I+\beta E)\\ &= I + \beta E + \frac{\gamma}{\alpha - \gamma w^\top(I+\beta E)u}(I+\beta E)uw^\top(I+\beta E).\end{aligned}$$

Now, observe that we have (see (20b))

$$\begin{aligned}(I+\beta E)u &= \frac{1}{\gamma}\big(T_k - c_k + \beta E(T_k - c_k - v_k)\big),\\ w^\top(I+\beta E) &= nv_k^\top\big(I - \frac{1}{n}E\big),\\ \alpha - \gamma w^\top(I+\beta E)u &= nv_k^\top(y_k + z_k).\end{aligned}$$

Therefore,

$$(I - \gamma\widetilde{P}_k)^{-1} = I + \beta E + \frac{\big(T_k - c_k + \beta E(T_k - c_k - v_k)\big)v_k^\top\big(I - \frac{1}{n}E\big)}{v_k^\top(y_k+z_k)},$$

and, hence, for the update rule (14), we have (see (20b))

$$\begin{aligned}v_{k+1} &= v_k - (I-\gamma\widetilde{P}_k)^{-1}(v_k - T_k) = T_k - \beta E g_k - \frac{\big(T_k - c_k + \beta E(-c_k - g_k)\big)v_k^\top y_k}{v_k^\top(y_k+z_k)}\\ &= T_k - \beta E g_k - \delta_k\big(T_k - c_k - \beta E(c_k + g_k)\big) = (1-\delta_k)T_k + \delta_k c_k + \beta E\big(-g_k + \delta_k(c_k+g_k)\big)\\ &= (1-\delta_k)T_k + \delta_k c_k + \beta\Big(\mathbf{1}^\top\big((\delta_k - 1)g_k + \delta_k c_k\big)\Big)\mathbf{1} = (1-\delta_k)T_k + \delta_k c_k + \lambda_k\mathbf{1}.\end{aligned}$$

For the case $v_k^\top(y_k+z_k) = 0$, as we discussed in the beginning of the proof, the two equality constraints in the minimization problem (18) become linearly dependent, and, in particular, the second constraint can be discarded. The solution to the problem (18) in this case is then $\widetilde{P}_k = \frac{1}{n}E$ and hence the update rule (14) can be written as (recall that $\delta_k = 0$)

$$\begin{aligned}v_{k+1} &= v_k - \big(I - \frac{\gamma}{n}E\big)^{-1}(v_k - T_k) = v_k - (I+\beta E)(v_k - T_k) = T_k - \beta(e^\top g_k)\mathbf{1}\\ &= (1-\delta_k)T_k + \delta_k c_k + \lambda_k\mathbf{1}.\end{aligned}$$

Next, we consider the rate of convergence of the safeguarded QPI update rule. Observe that since T is a γ-contraction in ∞-norm, the safeguarding using standard VI as in (19) implies that

$$\|v_{k+1} - T_{k+1}\|_\infty \le \max\{\gamma^{k+1}\|v_0 - T_0\|_\infty,\ \gamma\|v_k - T_k\|_\infty\},$$

for all $k \ge 0$. This ensures a linear convergence with rate γ.

Finally, the per-iteration time complexity of each iteration of the safeguarded QPI update rule: The update rule (20) requires $\mathcal{O}(n^2m)$ operations for computing the vectors $T_k = T(v_k)$ and c_k, and $\mathcal{O}(n)$ operations for computing the step-sizes δ_k and λ_k and the vector additions. For the safeguarding (19), we need to compute $T_{k+1} = T(v_{k+1})$ which again requires $\mathcal{O}(n^2m)$ operations. Summing up the aforementioned complexities, we derive the total time complexity to be $\mathcal{O}(n^2m)$.

A.3. **Proof of Theorem 4.3**

The per-iteration time complexity of each iteration of the safeguarded QPL update rule: The update rule (21) requires $\mathcal{O}(nm^2)$ operations for computing the vectors $\widehat{T}_k = \widehat{T}(q_k, \hat{s}_k^+)$, and $\mathcal{O}(nm)$ operations for computing the step-sizes δ_k and λ_k and the vector additions. For the safeguarding (22), we need to run a QL algorithm in parallel which also has an $\mathcal{O}(nm^2)$ per-iteration complexity. Moreover, we need to compute $\widehat{T}_{k+1}$ (for computing $\widehat{\theta}_{k+1}$) which again requires $\mathcal{O}(n^2m)$ operations. Summing up the aforementioned complexities, the total time complexity is $\mathcal{O}(nm^2)$.

Regarding the convergence, observe that the safeguarding ensures that the accumulated sampled Bellman for QPL is dominated by that of QL ran in parallel, that is, $\sum_{i=0}^{k} \widehat{\theta}_i \leq \sum_{i=0}^{k} \widehat{\theta}_i^{\mathrm{QL}}$ for all $k > K_{\mathrm{sg}}$, and, if not, it replaces the QPL update by a QL update. This ensures the convergence of QPL with at least the same rate as QL.

Appendix B. **QPI and QPL Algorithms**

Algorithm 1 provides the pseudo-code of the safeguarded QPI algorithm with arbitrary initialization v_0. We note that the output of Algorithm 1 satisfies $\|v^\epsilon - v^\star\|_\infty \leq \epsilon/(1-\gamma)$, where $v^\star$ is the optimal value function. The pseudo-code for the safeguarded QPL algorithm with arbitrary initialization q_0 is provided in Algorithm 2. We note that lines 4 and 13 of Algorithm 2 are related to the QL algorithm ran in parallel for the proposed safeguarding.

Algorithm 1 Quasi-Policy Iteration (QPI)

Input: cost $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; probability kernel $\mathbb{P}$; discount factor $\gamma \in (0,1)$; termination constant $\epsilon > 0$;
Output: sub-optimal value function v^ϵ;

```
1: initialize $v_0 \in \mathbb{R}^n$;
2: compute $T_0 = T(v_0)$ and $c_0 = c^{\pi_{v_0}}$; $\theta_0 = \|v_0 - T_0\|_\infty$;
3: for $k = 0, 1, 2, \dots$ do
4:     if $\|v_k - T_k\|_\infty \leq \epsilon$, then terminate;
5:     $z_k = c_k - \frac{\mathbf{1}^\top c_k}{n}\mathbf{1}$; $g_k = v_k - T_k$; $y_k = g_k - \frac{\mathbf{1}^\top g_k}{n}\mathbf{1}$;
6:     $\delta_k = \frac{v_k^\top y_k}{v_k^\top (y_k + z_k)}$ if $v_k^\top (y_k + z_k) \neq 0$, and $= 0$ otherwise; $\lambda_k = \frac{\gamma}{n(1-\gamma)}\mathbf{1}^\top\big((\delta_k - 1)g_k + \delta_k c_k\big)$;
7:     $v_{k+1} = (1-\delta_k)T_k + \delta_k c_k + \lambda_k \mathbf{1}$;
8:     compute $T_{k+1} = T(v_{k+1})$ and $c_{k+1} = c^{\pi_{v_{k+1}}}$;
9:     if $\|v_{k+1} - T_{k+1}\|_\infty > \gamma^{k+1}\theta_0$, then safeguard:
                  $v_{k+1} = T_k$; recompute $T_{k+1} = T(v_{k+1})$ and $c_{k+1} = c^{\pi_{v_{k+1}}}$;
10: end for
11: $v^\epsilon = v_k$;
```

Algorithm 2 Quasi-Policy Learning (QPL)

```
Input: cost $c \in \mathbb{R}^{nm}$; discount factor $\gamma \in (0,1)$; maximum number $K$ of iterations; safeguard activation
    iteration number $K_{\rm sg}$;
Output: sub-optimal Q-function $q^{\rm out}$;
 1: initialize $q_0 \in \mathbb{R}^{nm}$;
 2: generate samples $\hat{s}_0^+ \sim \mathbb{P}(\cdot|s,a)$ for each $(s,a) \in \mathcal{S} \times \mathcal{A}$;
 3: compute $\widehat{T}_0 = [\widehat{T}(q_0, \hat{s}_0^+)]$; $\widehat{\Theta}_0 = \|q_0 - \widehat{T}_0\|_\infty$;
 4: $q_0^{\rm QL} = q_0$; $\widehat{T}_0^{\rm QL} = \widehat{T}_0$; $\widehat{\Theta}_0^{\rm QL} = \widehat{\Theta}_0$;
 5: $z = c - \frac{\mathbf{1}^\top c}{nm}\mathbf{1}$;
 6: for $k = 0, 1, 2, \dots, K-1$ do
 7:     $\alpha_k = 1/(k+1)$;
 8:     $\widehat{g}_k = q_k - \widehat{T}_k$; $y_k = \widehat{g}_k - \frac{\mathbf{1}^\top \widehat{g}_k}{nm}\mathbf{1}$;
 9:     $\delta_k = \frac{q_k^\top y_k}{q_k^\top (y_k + z)}$ if $q_k^\top (y_k + z) \neq 0$, $= 0$ otherwise; $\lambda_k = \frac{\gamma}{nm(1-\gamma)}\mathbf{1}^\top\big((\delta_k - 1)\widehat{g}_k + \delta_k c\big)$;
10:     $q_{k+1} = (1-\alpha_k)q_k + \alpha_k\big((1-\delta_k)\widehat{T}_k + \delta_k c + \lambda_k \mathbf{1}\big)$;
11:     generate samples $\hat{s}_{k+1}^+ \sim \mathbb{P}(\cdot|s,a)$ for each $(s,a) \in \mathcal{S} \times \mathcal{A}$;
12:     compute $\widehat{T}_{k+1} = [\widehat{T}(q_{k+1}, \hat{s}_{k+1}^+)]$; $\widehat{\theta}_{k+1} = \|q_{k+1} - \widehat{T}_{k+1}\|_\infty$;
13:     $q_{k+1}^{\rm QL} = (1-\alpha_k)q_k^{\rm QL} + \alpha_k \widehat{T}_k^{\rm QL}$; compute $\widehat{T}_{k+1}^{\rm QL} = [\widehat{T}(q_{k+1}^{\rm QL}, \hat{s}_{k+1}^+)]$; $\widehat{\Theta}_{k+1}^{\rm QL} = \widehat{\Theta}_k^{\rm QL} + \|q_{k+1}^{\rm QL} - \widehat{T}_{k+1}^{\rm QL}\|_\infty$;
14:     if $k > K_{\rm sg}$ and $\widehat{\Theta}_k + \widehat{\theta}_{k+1} > \widehat{\Theta}_{k+1}^{\rm QL}$, then safeguard:
                $q_{k+1} = (1-\alpha_k)q_k + \alpha_k \widehat{T}_k$; recompute $\widehat{T}_{k+1} = [\widehat{T}(q_{k+1}, \hat{s}_{k+1}^+)]$; $\widehat{\theta}_{k+1} = \|q_{k+1} - \widehat{T}_{k+1}\|_\infty$;
15:     $\widehat{\Theta}_{k+1} = \widehat{\Theta}_k + \widehat{\theta}_{k+1}$;
16: end for
17: $q^{\rm out} = q_K$;
```

Appendix C. Numerical Simulations

C.1. Garnet MDPs

The control problem instances in this study are based on randomly generated Garnet MDPs [3]with $n = 50$ states, $m = 5$ actions, and the branching parameter $n_b = 10$. For each state-action pair (s,a), we first form the set of reachable next states $\{s_1^+, \dots, s_{n_b}^+\}$ chosen uniformly at random from the state space $\{1, \dots, n\}$. Then, the corresponding probabilities are formed by choosing the points $p_i \in [0,1], i = 1, \dots, n_b - 1$, uniformly at random, and setting $\mathbb{P}(s_i^+|s,a) = p_i - p_{i-1}$ with $p_0 = 0$ and $p_{n_b} = 1$. The stage cost $c(s,a)$ for each state-action pair (s,a) is also chosen uniformly at random from the interval $[0,1]$.

C.2. Nesterov accelerated VI (NVI) algorithm[19]

The NVI algorithm is based on the update rule (with initialization $v_{-1} = v_0 = \mathbf{0}$)

$$\begin{aligned} y_k &= v_k + \frac{1 - \sqrt{1-\gamma^2}}{\gamma}(v_k - v_{k-1}), \\ v_{k+1} &= y_k - \frac{1}{1+\gamma}\big(y_k - T(y_k)\big). \end{aligned}$$

C.3. **Anderson accelerated VI (AVI) algorithm** [16]

The AVI algorithm is based on the update rule (with initialization $v_{-1} = v_0 = \mathbf{0}$)

$$\begin{aligned}
y_k &= v_k - v_{k-1}, \\
z_k &= T(v_k) - T(v_{k-1}), \\
\delta_k &= \begin{cases} 0, & y_k^\top (y_k - z_k) = 0, \\ \frac{y_k^\top \left(v_k - T(v_k)\right)}{y_k^\top (y_k - z_k)}, & \text{otherwise}, \end{cases} \\
v_{k+1} &= (1-\delta_k) T(v_k) + \delta_k T(v_{k-1}).
\end{aligned}$$

References

[1] Allen-Zhu, Z. (2017). Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244.

[2] Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560.

[3] Archibald, T., McKinnon, K., and Thomas, L. (1995). On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46(3):354–361.

[4] Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

[5] Berthier, E. and Bach, F. (2020). Max-plus linear approximations for deterministic continuous-state markov decision processes. *IEEE Control Systems Letters*, 4(3):767–772.

[6] Bertsekas, D. (1975). Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419.

[7] Bertsekas, D. (2022). *Abstract dynamic programming*. Athena Scientific.

[8] Bertsekas, D. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

[9] Bertsekas, D. P. (2011). Temporal difference methods for general projected equations. *IEEE Transactions on Automatic Control*, 56(9):2128–2139.

[10] Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

[11] De Farias, D. P. and Van Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478.

[12] Devraj, A. M., Bušić, A., and Meyn, S. (2019). On matrix momentum stochastic approximation and applications to Q-learning. In *57th Annual Allerton Conference on Communication, Control, and Computing*, pages 749–756.

[13] Devraj, A. M. and Meyn, S. (2017). Zap Q-learning. In *Advances in Neural Information Processing Systems*, volume 30.

[14] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

[15] Gargiani, M., Zanelli, A., Liao-McPherson, D., Summers, T., and Lygeros, J. (2022). Dynamic programming through the lens of semismooth Newton-type methods. *IEEE Control Systems Letters*, 6:2996–3001.

[16] Geist, M. and Scherrer, B. (2018). Anderson acceleration for reinforcement learning. *preprint arXiv:1809.09501*.

[17] Ghavamzadeh, M., Kappen, H., Azar, M., and Munos, R. (2011). Speedy Q-learning. In *Advances in Neural Information Processing Systems*, volume 24.

[18] Gonçalves, V. M. (2021). Max-plus approximation for reinforcement learning. *Automatica*, 129:109623.

[19] Goyal, V. and Grand-Clément, J. (2022). A first-order approach to accelerated value iteration. *Operations Research*, 71(2):517–535.

[20] Grand-Clément, J. (2021). From convex optimization to MDPs: A review of first-order, second-order and quasi-Newton methods for MDPs. *preprint arXiv:2104.10677*.

[21] Hernández-Lerma, O. and Lasserre, J. B. (2012a). *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media.

[22] Hernández-Lerma, O. and Lasserre, J. B. (2012b). *Further topics on discrete-time Markov control processes*, volume 42. Springer Science & Business Media.

[23] Howard, R. A. (1960). *Dynamic programming and Markov processes.* John Wiley.

[24] Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *19th International Conference on Machine Learning*, pages 267–274.

[25] Kamanchi, C., Diddigi, R. B., and Bhatnagar, S. (2022). Generalized second order value iteration in Markov decision processes. *IEEE Transactions on Automatic Control*, 67(8):4241–4247.

[26] Kearns, M. and Singh, S. (1998). Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in neural information processing systems*, volume 11.

[27] Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. (2018). On the insufficiency of existing momentum schemes for stochastic optimization. In *Information Theory and Applications Workshop*, pages 1–9.

[28] Kolarijani, M. A. S., Max, G. F., and Mohajerin Esfahani, P. (2021). Fast approximate dynamic programming for infinite-horizon markov decision processes. In *Advances in Neural Information Processing Systems*, volume 34, pages 23652–23663.

[29] Kolarijani, M. A. S. and Mohajerin Esfahani, P. (2023). Fast approximate dynamic programming for input-affine dynamics. *IEEE Transactions on Automatic Control*, 68(10):6315–6322.

[30] Kushner, H. and Kleinman, A. (1971). Accelerated procedures for the solution of discrete Markov control problems. *IEEE Transactions on Automatic Control*, 16(2):147–152.

[31] Lemaréchal, C. (2012). Cauchy and the gradient method. *Documenta Mathematica Extra*, pages 251–254.

[32] Liu, C. and Belkin, M. (2018). Accelerating SGD with momentum for over-parameterized learning. *preprint arXiv:1810.13395*.

[33] Liu, J. and Yuan, Y. (2022). On almost sure convergence rates of stochastic gradient methods. In *35th Conference on Learning Theory*, pages 2963–2983.

[34] McEneaney, W. M. (2006). *Max-plus methods for nonlinear control and estimation.* Springer Science & Business Media.

[35] Mohajerin Esfahani, P., Sutter, T., Kuhn, D., and Lygeros, J. (2018). From infinite to finite programs: Explicit error bounds with applications to approximate dynamic programming. *SIAM Journal on Optimization*, 28(3):1968–1998.

[36] Nesterov, Y. (2018). *Lectures on convex optimization.* Springer.

[37] Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Doklady Akademii Nauk SSSR*, volume 269, pages 543–547.

[38] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

[39] Porteus, E. L. and Totten, J. C. (1978). Accelerated computation of the expected discounted return in a Markov chain. *Operations Research*, 26(2):350–358.

[40] Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

[41] Puterman, M. L. and Brumelle, S. L. (1979). On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4(1):60–69.

[42] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

[43] Ruppert, D. (1985). A Newton-Raphson version of the multivariate Robbins-Monro procedure. *The Annals of Statistics*, 13(1):236–245.

[44] Rust, J. (1994). Structural estimation of Markov decision processes. *Handbook of Econometrics*, 4:3081–3143.

[45] Samuelson, P. A. (1948). Foundations of economic analysis. *Science and Society*, 13(1).

[46] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[47] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

[48] Szepesvári, C. (2010). *Algorithms for reinforcement learning.* Morgan & Claypool.

[49] Tsitsiklis, J. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

[50] Vieillard, N., Pietquin, O., and Geist, M. (2019). On connections between constrained optimization and reinforcement learning. *preprint arXiv:1910.08476.*

[51] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[52] Weng, B., Xiong, H., Zhao, L., Liang, Y., and Zhang, W. (2021). Finite-time theory for momentum Q-learning. In *37th Conference on Uncertainty in Artificial Intelligence*, pages 665–674.

[53] Yang, T., Lin, Q., and Li, Z. (2016). Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *preprint arXiv:1604.03257.*

[54] Zhang, J., O'Donoghue, B., and Boyd, S. (2020). Globally convergent type-I Anderson acceleration for nonsmooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4):3170–3197.